ECE 375 LAB 2

C →Assembler → Machine Code → Tekbot

Lab Time: Friday 4 - 6

Faaiq Waqar

Jordan Brown

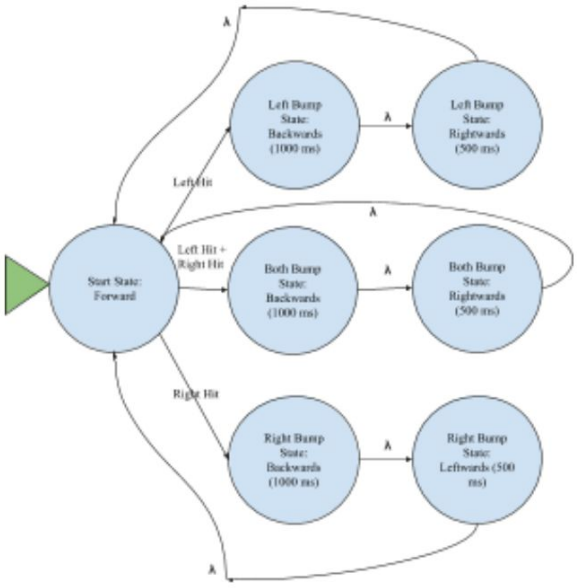
INTRODUCTION

The purpose of the second lab in ECE375 is to introduce control of I/O ports of the ATmega128 microcontroller through the use of a high level programming language, C. An example program known as “DanceBot.c” is provided to the user, which uses the I/O ports in order to manipulate a connected TekBot to move in a fashion that reflects “dancing”, by using certain port configurations to move the TekBot. This code can be used for the student to understand the format of port manipulation, and thus, the student in this lab will construct a program that in similar function, will move the robot, based on input received from the left and right whiskers. Once constructed, the project is compiled in AVRStudio4, built and tested on the AVR board

PROGRAM OVERVIEW

The C Version of the basic BumpBot routine will provide instruction to the TekBot to react to input from the whiskers, in this case will be signified by buttons 1 & 0 on the ATmega128. The TekBot has two forward facing buttons or whiskers, distinguished as the right whisker, and the left whisker. By default, the TekBot will be moving forward, done by the bot having both motors moving forward. If the left whisker is triggered, the TekBot will move backwards, then turn in the opposite direction (right) for 500 ms. Then the bot will continue to move forward, returning to its initial state. If the right whisker is hit, the TekBot will move backwards, then turn in the opposite direction (left), and then resuming moving forward as it did in its original state. If both whiskers are triggered at the same time, the program will use the same protocol as a left whisker bump.

There are two primary portions of the code, that contained in the loop, which will be the control state of the program, and that contained outside of the loop, which will be the initialization state of the program. In the initialization state of the program, the data direction registers are configured for B and D, allowing for appropriate input and output, specified to the directed pins. After this, the PORT registers are configured with initialization values for B and D. In the control state of the program, PORT B is set to forward, with branching statements used afterwards in cases of whisker triggering for the three different states. Below is a state diagram used to visualize the program execution.



Initialization Routine

The initialization routine provides activation of the data direction registers (DDRx) and port registers (PORTx). The data direction on B, which routes to the motors of the tekbot are mapped to output, or 1 on DDRB(7-4). This is how we will be able to receive data controlled into the motors via this I/O port. For the pin output register signified by PORTB, 1 was mapped to PORTB(6-5). This determined the high or low on output configuration, and the config set moves the TekBot into active forward. On the data direction side of D, or whiskers, DDRD(1-0) are set into output (1) configuration. We also configure PORTD(1-0) to 1, setting them as default off signals due to the typical active low status of port D.

Control Routine

In the control routine, an infinite loop with no exit condition is used with a default pin output on B set to active on PORTB(6-5), indicating a default forward position unless a separate state condition will cause the program to move into one of the three whisker trigger states. This will loop infinitely unless the operating system or the system hardware either sends a SIGINT, or the hardware is shut off.

RightTrigger Routine

There are two instances of the RightTrigger routine, depending on the execution of the procedure code or the challenge code. This portion will discuss the procedure code. There are two primary operations to be conducted in this section. The first instance is the backup phase. The pin output is set to PORTB(7-0) equal to 0, indicating a backwards movement on both motors. A delay function is used then for a parameter of 1000, indicating a 1000 millisecond delay before stepping into the next action/state. The next state will shift the TekBot into a left turn state, meaning that the pin output will be set to PORTB(5) on (1). This means that the left hand motor will go in the reverse direction and the right motor will go in the forwards direction, causing a left shift. This will continue due to a delay function use for 500 ms, before returning to the initial state once again.

In the instance of the program for challenge code, the state will instead begin with a delay of 500 ms, indicating an attempt to push forward on the object for a specified amount of time. From here the state will shift into 1000 ms backwards movement through the pin output set to PORTB(7-0) equal to 0. After this the pin output will be set to PORTB(6) on (1). This means that the left hand motor will go in the forward direction and the right motor will go in the reverse direction, causing a right shift. This will continue due to a delay function use for 500 ms, before returning to the initial state once again.

LeftTrigger Routine

There are two instances of the LeftTrigger routine, depending on the execution of the procedure code or the challenge code. This portion will discuss the procedure code. There are two primary operations to be conducted in this section. The first instance is the backup phase. The pin output is set to PORTB(7-0) equal to 0, indicating a backwards movement on both motors. A delay function is used then for a parameter of 1000, indicating a 1000 millisecond delay before stepping into the next action/state. The next state will shift the TekBot into a right turn state, meaning that the pin output will be set to PORTB(6) on (1). This means that the left hand motor will go in the forward direction and the right motor will go in the reverse direction, causing a right shift. This will continue due to a delay function use for 500 ms, before returning to the initial state once again.

In the instance of the program for challenge code, the state will instead begin with a delay of 500 ms, indicating an attempt to push forward on the object for a specified amount of time. From here the state will shift into 1000 ms

backwards movement through the pin output set to PORTB(7-0) equal to 0. After this the pin output will be set to PORTB(5) on (1). This means that the left hand motor will go in the reverse direction and the right motor will go in the forward direction, causing a left shift. This will continue due to a delay function use for 500 ms, before returning to the initial state once again.

BothTrigger Routine

If both whiskers are triggered at the same time, in the case of the procedure code and the challenge code, the program will work under the same instruction set as a left whisker bump, meaning that the same states will be followed for this portion of the program.

Additional Questions

1. This lab required you to compile two C programs (one given as a sample, and another that you wrote) into a binary representation that allows them to run directly on your mega128 board. Explain some of the benefits of writing code in a language like C that can be "cross compiled". Also, explain some of the drawbacks of writing this way

   Cross compiling allows code to run on any platform and not be for a specific machine. For example, our C code written for lab 2 could be cross compiled to work with several machines that have different kinds of architectures. Writing code to be cross compiled can increase development time and product testing time. Making code that works with each compiler can be more time consuming because code can be written that depends on compiler specific behavior.

2. The C program you just wrote does basically the same thing as the sample assembly program you looked at in Lab 1. What is the size (in bytes) of your Lab 1 & Lab 2 output .hex files? Can you explain why there is a size difference between these two files, even though they both perform the same BumpBot behavior

   Lab 2 was 2 KB while lab 1 was only 1 KB. Despite being significantly more lines, Lab 1 was able to have a smaller executable file while performing the same task. Writing in C might improve readability, but assembly allows the programmer to save important program memory when working with limited storage. For these first 2 labs this difference in size didn't matter but when using a micro controller for real world applications that require efficient use of all the program memory, assembly language is the best choice.

Difficulties

Our understanding of the task was more than sufficient but we were unfamiliar with how to use the I/O registers. We were having issues communicating with the motors or LEDs which occurred because we didn't initialize PORTB. Our next issue was a result of us using PORT instead of pin when writing to our binary strings to the motors. With both of these issues resolved our code ran well and accomplished the task.

Conclusion

This lab was an introduction into using the I/O registers on the atmega128. We were reading information from 2 buttons on the front of the bot and writing out to the motors depending on the input. The bots behavior was controlled using conditionals that checked for different configurations of inputs from the whiskers. This program

was written in C, compiled using Atmel, and uploaded to the board. We had a few difficulties with the lab, but overcoming them was very beneficial to our learning.

Source Code

```
// Faaiq_Waqar_Jordan_Brown_Lab2_sourcecode.c

/*

This code will cause a TekBot connected to the AVR board to

move forward and when it touches an obstacle, it will reverse

and turn away from the obstacle and resume forward motion.


PORT MAP

Port B, Pin 4 -> Output -> Right Motor Enable

Port B, Pin 5 -> Output -> Right Motor Direction

Port B, Pin 7 -> Output -> Left Motor Enable

Port B, Pin 6 -> Output -> Left Motor Direction

Port D, Pin 1 -> Input -> Left Whisker

Port D, Pin 0 -> Input -> Right Whisker

*/


/*

AUTHOR(S): Faaiq Waqar & Jordan Brown

DATE: October 11th 2019

COURSE AND LAB: ECE 375 :: Friday 4-6

VERSION: Challenge Program Code

*/


#define F_CPU 16000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>
```

```
int main(void)

{

  DDRB = 0b11110000; //configure port B pins for input/output

  DDRD = 0b00000011; //Configure port D pins for input/output

  PORTB = 0b01100000; //Set initial value for Port B outputs

  PORTD = 0b00000011; //Set initial value for port D outputs


while (1) // loop forever

      {

        PORTB = 0b01100000; //Set initial forward state in loop

        if(PORTD == 0b00000010){ //Right Whisker INdicator

          PORTB = 0b00000000; //reverse for a bit

          _delay_ms(1000); //delary for 1000 ms

          PORTB = 0b0100000; //turn away from obj

          _delay_ms(500); //do the turning away for 500 ms

        }

        else if(PORTD == 0b00000001){ //Left Whisker

          PORTB = 0b00000000; //reverse for 1000 ms

          _delay_ms(1000);

          PORTB = 0b0010000; //turn away for 500 ms

          _delay_ms(500);

        }

        else if(PORTD == 0b00000000){ //Both Whisker

          PORTB = 0b00000000; //copy the section for left whisker

          _delay_ms(1000);

          PORTB = 0b0010000;

          _delay_ms(500);

        }

            // Your code goes here

      }
```

```
}
```

Challenge Code

```
// Faaiq_Waqar_Jordan_Brown_Lab2_challengecode.c

/*

This code will cause a TekBot connected to the AVR board to

move forward and when it touches an obstacle, it will reverse

and turn away from the obstacle and resume forward motion.

PORT MAP

Port B, Pin 4 -> Output -> Right Motor Enable

Port B, Pin 5 -> Output -> Right Motor Direction

Port B, Pin 7 -> Output -> Left Motor Enable

Port B, Pin 6 -> Output -> Left Motor Direction

Port D, Pin 1 -> Input -> Left Whisker

Port D, Pin 0 -> Input -> Right Whisker

*/


/*

AUTHOR(S): Faaiq Waqar & Jordan Brown

DATE: October 11th 2019

COURSE AND LAB: ECE 375 :: Friday 4-6

VERSION: Challenge Program Code

*/


#define F_CPU 16000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>


int main(void)
```

```
{
        DDRB = 0b11110000; //configure port B pins for input/output
        DDRD = 0b00000000; //Configure port D pins for input/output
        PORTB = 0b11110000;
        PORTD = 0b00000011;


        while (1) // loop forever
        {
                PORTB = 0b01100000;

                if(PIND == 0b11111101){ //Right Whisker
                        _delay_ms(500);//wait while continuing forward
                        PORTB = 0b00000000;//reverse
                        _delay_ms(1000); //wait 1 second
                        PORTB = 0b00100000;// turn right toward object
                        _delay_ms(500);//wait then exit conditional and continue forward
                }

                else if(PIND == 0b11111110){ //Left Whiskey
                        _delay_ms(500);//wait while continuing forward
                        PORTB = 0b00000000;//reverse
                        _delay_ms(1000);//wait 1 second
                        PORTB = 0b01000000; //turn left toward the object
                        _delay_ms(500);//wait then exit conditional and continue forward
                }

                else if(PIND == 0b11111100){ //Both WHisker
                        _delay_ms(500);//wait while continuing forward
                        PORTB = 0b00000000;//reverse
```

```
                    _delay_ms(1000);//wait  then  exit  the  conditional  and  begin  moving
forward
                    //PORTB = 0b01000000;
                    //_delay_ms(500);


                }









        }
}
```